CXQuery 를 이용한 XML 스트리밍 데이터 필터링

김소라*, 이월영**, 용환승*
*이화여자대학교 컴퓨터학과
**서울기독대학교 국제경영정보학과
e-mail : sollar00@ewhain.net

XML Streaming Data Filtering using CXQuery

So-Ra Kim*, Wol-young Lee**, Hwan-Seung Yong*
*Dept. of Computer Science, Ewha Woman's University
**Dept. of International Business Information, Seoul Christian University

요 약

XML 은 환경에 독립적인 특징으로 인하여 다양한 환경에서 데이터 교환을 위한 표준으로서 자리잡아가고 있다. 특히 분산 환경에서 스트림 데이터들도 XML 을 이용하여 표현되고 있는데 이러한 데이터에 대해 기존의 질의 언어를 사용하여 질의하기 위해서는 사용자들은 XML 문서의 구조를 알아야 하고, 같은 질의의 내용일지라도 XML 문서의 DTD 가 다르게 되면 사용자는 상이한 DTD 에 대해서 모두 다른 질의문을 작성하여야 한다. 이러한 사용자들의 불편함을 없애기 위하여 사용자에게 XML 문서의 구조에 대해서 고려 하지 않아도 검색할 수 있는 CXQuery 의 개념을 도입하여 스트리밍 XML 데이터를 효과적으로 처리할 수 있는 질의 처리 시스템을 구현하였다.

1. 서론

XML 은 쉽고 일관된 방법으로 데이터를 포맷하고 전송하는데 사용한다. XML 은 그 유연성 때문에 웹 상에서 데이터와 각종 문서 교환을 위한 표준으로 간주되고 있다. 웹 서비스와 같은 분산 컴퓨팅 환경에서 위와 같은 이유로 데이터들은 XML 로 인코딩되어 교환되고 있고, 이에 따라 XML 데이터 필터링에 대한 연구가 활발히 이루어지고 있다. XML 필터링 시스템에서는 조건에 맞는 데이터들을 추출하기 위하여 XML 문서들이 스트림 형태로 연속적으로 필터링 엔진으로 들어간다.

이 때 기존의 질의 언어[7,8]를 사용하여 원하는 데이터를 추출하기 위해서 사용자는 그 문서의 구조를 알아야 한다. 이러한 사용자의 불편함을 없애기 위해 CXQuery 라는 새로운 질의 언어가 개발되었다[1]. 그러나 이 질의 언어에서 제공하고 있는 질의 처리 기법은 XML 문서를 파싱하여 기존의 데이터베이스에 저장한 후 질의 처리하기 때문에, 데이터를 저장하고 처리하기에는 너무 방대할 뿐 아니라 연속적으로 변화하는 스트림 데이터에 대해서는 적용하기가 어렵다.

따라서 본 연구에서는 질의 표현식에서는 문서 구조에 독립적인 CXQuery 의 개념을 도입하고 이를 위한 질의 처리기는 스트림 데이터에 효과적인 기법을 개발한다.

본 논문의 구성은 다음과 같다. 제 2 장의 관련연구로 이 시스템에 도입된 기본 개념인 CXQuery 와 XML 필터링 시스템 들에 대한 관련 연구를 소개할 것이고, 3 장에서는 이 논문에서 제시하고 있는 시스템에 대한 소개를 할 것이다. 4 장에서 이 시스템을 바탕으로 한 실험평가에 대해서 서술한 다음 마지막 장에서는 결론을 낼 것이다.

* 본 연구는 한국과학재단 목적기초연구(R01-2003-000-10395-0)지원으로 수행되었음

2. 관련연구

방대한 양의 XML 문서를 효율적으로 필터링하도록 하는 시스템에 대한 연구가 많이 있어 왔다. 그 중에서 대표적인 것이 XFilter, YFilter 이다. XFilter 는 스트림 XML 데이터를 필터링하는데 있어 XPath 형식으로 표현된 사용자의 프로파일에 기반하여 XML 문서를 필터링하는 시스템으로, 구조 기반의 XML 필터링을 위해 경로로 표현된 질의를 FSM(Finite State Machines)를 사용한다. XFilter 는 각각의 모든 질의에 대해 새로운 FSM 을 만들어내기 때문에 경로 표현에 있어 공통되는 부분에 대해서 중복적인 작업을 하게 된다.

이러한 XFilter 의 단점을 보완하기 위해 행해진 연구가 YFilter 이다. YFilter 는 경로를 NFA(Nondeter-ministic Finite Automata)으로 표현을 하여 XFilter 에서의 중복적인 작업을 없애고, NFA 를 사용함으로써 경로의 공통 부분을 공유하도록 하여 한 번만 처리되도록 한다. 이것은 XFilter 에 비해서 표현되는 상태의 수를 줄임으로써 효율적으로 필터링을 수행하도록 한다.

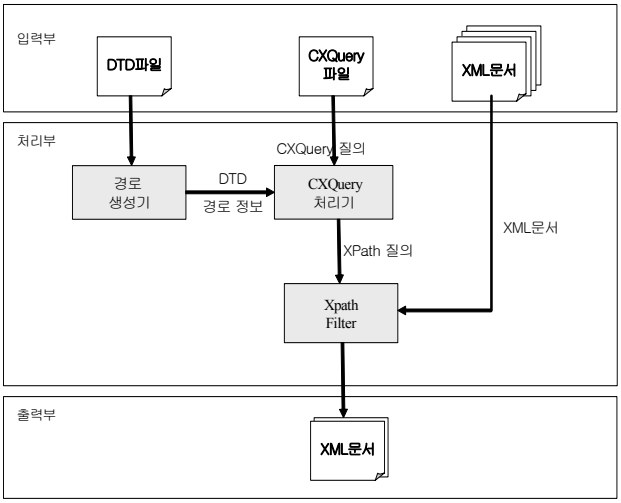
본 논문에서는 기존의 질의 언어가 문서 구조를 알아야 하는 사용자의 불편함을 없애기 위해 CXQuery 의 개념을 도입, 질의를 경로표현 대신에 데이터의 이름과 값만으로 표현하여 이를 YFilter 에 적용한 후 질의 결과를 얻어낼 수 있도록 할 것이다.

3. 설계 및 구현

이 시스템에서는 사용자가 XML 로 표현된 스트림 데이터를 처리하는데 있어서 XML 문서의 구조를 알지 못하더라도 편리하게 필터링을 할 수 있도록 하였다. 이를 위해 질의 처리기는 모든 XML 문서의 구조를 파악하여 사용자로부터 찾으려는 데이터 이름과 그 데이터의 값만을 입력 받더라도 그 데이터가 들어간 모든 가능한 경로를 찾아내 원하는 결과를 얻어낼 수 있도록 구성하였다.

3.1 질의 시스템 구조

스트림 데이터에 대한 필터링 시스템의 구조도는 (그림 1)과 같다. 이것은 입력, 처리, 출력으로 나누어 지는데 이 중 처리부는 크게 경로 생성기, CXQuery 처리기, XPath Filer 로 이루어진다



(그림 1) 시스템 구조도

시스템의 동작 과정에 대해서 살펴보면 다음과 같다.

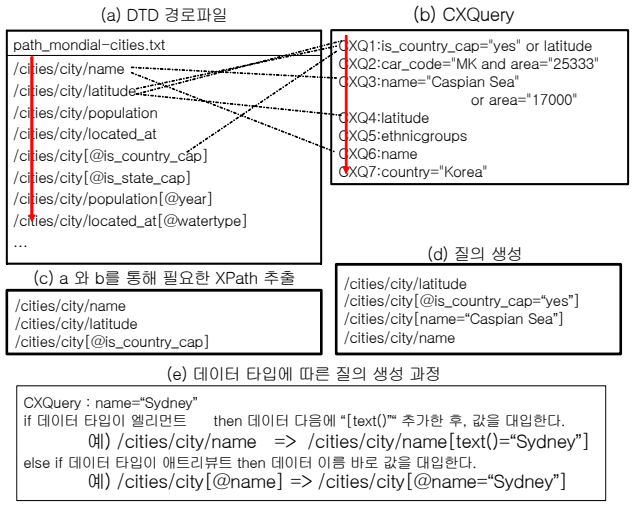
- 시스템의 입력값으로 XML 문서들과 CXQuery 가 들어간다.
- CXQuery 처리기는 CXQuery 파일을 XPath 필터기의 질의입력 형태인 XPath 질의 파일로 변환한다.
- 필터기는 XML 문서들과 XPath 질의 파일을 이용하여 필터링한다.

3. 2 질의 처리기

질의 시스템에서 가장 핵심인 질의 처리 부분은 XPath Filter, CXQuery 처리기, 경로 생성기로 나누어진다.

3.2.1 CXQuery 처리기

이는 입력 XML 문서의 경로 파일과 CXQuery 질의 파일을 가지고 CXQuery 파일을 XPath 질의 파일로 변환해 주는 부분이다. XML 문서의 경로파일이 있으면 그것들을 차례대로 읽어나가면서, CXQuery 파일에 명시된 데이터 이름이 존재하는 경로들을 추출해 낸다. 만약 질의문에 데이터 값이 있는 경우 각 경로에 값을 넣도록 하여 그 조건에 맞는 결과를 찾아낼 수 있도록 한다. 이 내용을 xpath_*CXQuery 파일명*.txt 에 적는다. CXQuery 쿼리 파일 한 개당 한 개의 xpath_*CXQuery 파일명*.txt 을 생성하여 후에 같은 질의를 사용하고자 할 때 재사용될 수 있도록 한다.



(그림 2) CXQuery 처리기에서 처리과정

3.2.2 XPath 필터기

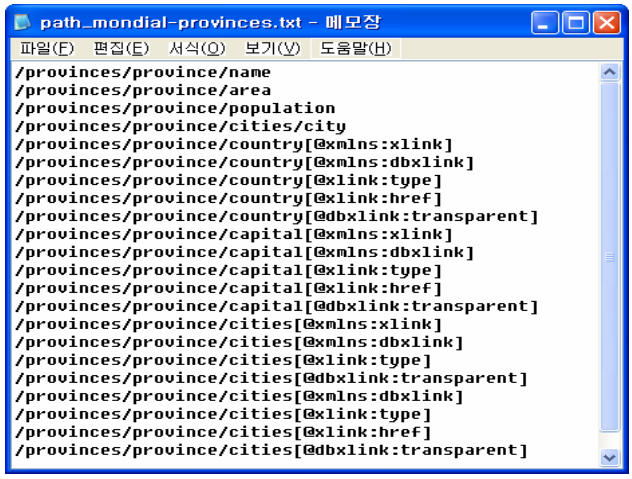
필터링 부분은 버클리 대학에서 제안한 기존의 YFilter 을 바탕으로 하여 구성된 부분이다.
전체 시스템의 입력 값은 처리하고자 하는 XML 문서들과 CXQuery 질의문이 저장되어 있는 질의 파일이다. CXQuery 처리기를 통해 XPath 질의 파일로 변환된 질의 파일이 생성되고, XPath 질의 파일과 XML 문서를 바탕으로 필터링을 수행한다.

3.2.3 경로 생성기

경로 생성기는 입력 받은 XML 의 DTD 를 파악, 각 DTD 에 대해 모든 가능한 경로들을 생성해내는 부분이다. CXQuery 처리기에서 CXQuery 를 XPath 로 된 질의 파일을 생성할 때 XML 문서의 경로파일이 필요하게 되는데, 이를 위해서 만든 컴포넌트이다. 각 DTD 의 가능한 모든 경로들은 path_*DTD 명*.txt 라는 경로 파일에 저장이 된다. 후에 XML 문서의 경로파일이 필요할 때, 기존의 경로파일을 재사용하게 된다.
(그림 3)는 예제 DTD 로 경로 생성기를 거치게 되면, (그림 4)과 같은 경로 파일이 생성된다.

```
<!-- XML DTD "mondial-provinces.dtd":  (Dimitrio Malheiro, malheiro@informatik.uni-freiburg.de, Juli 2002)
This DTD contains the description for provs-xxx.xml files -->
<!ELEMENT provinces (province+)>
<!ELEMENT province (name, country, capital*, area?, population, cities*)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT country EMPTY>
<!ELEMENT capital EMPTY>
<!ELEMENT area (#PCDATA)>
<!ELEMENT population (#PCDATA)>
<!ELEMENT cities (city*)>
<!ELEMENT city EMPTY>
<!ATTLIST country  xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink"
                   xmlns:dbxlink CDATA #FIXED "http://www.informatik.uni-freiburg.de/~malheiro/dbxlink"
                   xlink:type (simple|extended|locator|arc) #FIXED "simple"
                   xlink:href CDATA #IMPLIED
                   dbxlink:transparent NMTOKENS #IMPLIED>
<!ATTLIST capital  xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink"
                   xmlns:dbxlink CDATA #FIXED "http://www.informatik.uni-freiburg.de/~malheiro/dbxlink"
                   xlink:type (simple|extended|locator|arc) #FIXED "simple"
                   xlink:href CDATA #IMPLIED
                   dbxlink:transparent NMTOKENS #IMPLIED>
<!ATTLIST cities   xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink"
                   xmlns:dbxlink CDATA #FIXED http://www.informatik.uni-freiburg.de/~malheiro/dbxlink
                   xlink:type (simple|extended|locator|arc) #FIXED "extended"
                  dbxlink:transparent NMTOKENS #IMPLIED>
<!ATTLIST city     xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink"
                  xmlns:dbxlink CDATA #FIXED "http://www.informatik.uni-freiburg.de/~malheiro/dbxlink"
                  xlink:type (simple|extended|locator|arc) #FIXED "locator"
                  xlink:href CDATA #IMPLIED
                 dbxlink:transparent NMTOKENS #IMPLIED>
```

(그림 3) 예제 DTD



(그림 4) 위 DTD 의 경로 파일

4. 실험 및 평가 결과

이 시스템의 구현환경으로 시스템은 펜티엄 IV이며, 운영 체제는 윈도우 XP 이다. 프로그램은 JDK1.4 를 이용하여 Java 로 코딩하였다.

본 논문에서 XML 스트림 필터링 시스템의 성능을 측정하기 위해 두 가지의 경우에 대해서 실험하였다.

- C1: 사용자가 문서의 구조를 알고 XPath 로 질의를 하는 경우
- C2: 문서의 경로 파일들이 생성되지 않은 상태에서 CXQuery 을 사용하는 경우

C1 은 XPath 질의어를 이용하여 질의를 해서, 스트림 XML 문서를 필터링을 한 경우이고, C2 는 CXQuery 를 이용하여 질의를 한 경우이다.
C2 는 XPath 질의 파일을 생성해내기 위한 입력 XML 문서의 경로파일이 없는 경우로, XML 문서의 경로파일들을 생성해야 하고, 그것들을 바탕으로 XPath 질의 파일을 생성한다.
위의 두 가지의 경우에 대해 데이터의 양이 증가될 때 그 추이에 대해서 실험을 해 보았다. 다음 그래프는 실험의 결과이다.
실험에 사용된 CXQuery 질의파일 내용은 다음과 같다.

<표 1 >CXQuery 질의 파일

```
CXQ1:is_country_cap="yes" or latitude
CXQ2:car_code="MK and area="25333"
CXQ3:name="Caspian Sea" or area="17000"
CXQ4:latitude
CXQ5:ethnicgroups
CXQ6:name
```

<표 2> XPath 로 된 질의 파일

```
CXQ6:/cities/city/name
CXQ1:/cities/city/latitude
CXQ1:/cities/city[@is_country_cap= “yes” ]
CXQ3:/cities/city/name[text()= “Caspian Sea” ]
CXQ4:/cities/city/latitude
CXQ6:/continents/continent/name
CXQ6:/countries/country/name
```

```
CXQ5:/countries/country/ethnicgroups
CXQ3:/countries/country[@area="17000"]
CXQ5:/countries/country/ethnicgroups[@percentage]
CXQ6:/lakes/lake/name
```

<표 3> 실험 데이터

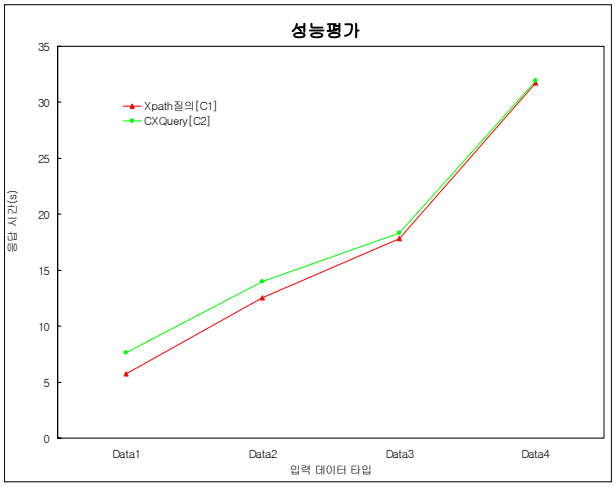
	Data1	Data2	Data3	Data4
XML 문서 개수(개)	100	200	300	400
데이터 크기(MB)	3.61	7.21	11.3	14.9
DTD1 개수	70	140	210	280
DTD2 개수	10	20	30	40
DTD3 개수	10	20	30	40
DTD4 개수	10	20	30	40

<표 4> 실험에 쓰인 DTD 데이터

	DTD1	DTD2	DTD3	DTD4
DTD 설명	도시정보 표현하는 XML 의 DTD	대륙정보 표현하는 XML 의 DTD	나라정보 표현하는 XML 의 DTD	강정보 표현하는 XML 의 DTD
엘리먼트개수	9	4	23	4
애트리뷰트개수	17	0	44	0

<표 5> 실험결과

	Data1	Data2	Data3	Data4
C1 걸린시간(s)	5.7	12.5	17.8	31.7
C2 걸린시간(s)	7.6	14	18.3	31.9



(그림 5) 입력 데이터에 따른 응답시간

위 실험 결과를 살펴보면 각각의 질의문에 대해서 결과를 보여주는 응답 시간은 큰 차이를 보이지 않는다. XPath 질의를 사용한 경우(C1)가 CXQuery 질의를 사용한 경우(C2)보다 응답 시간이 좀 짧은데, 이는 XML 문서의 경로 파일을 생성하고, CXQuery 를 바탕으로 XPath 질의로 만들어내는 과정이 필요 없기 때문이다.

5. 결론 및 향후 연구과제

본 논문에서는 XML 로 표현된 스트림 데이터에 대해 원하는 데이터를 검색할 때 문서 구조를 모르고서도 질의할 수 있고 원하는 결과를 얻을 수 있는 질의 처리 시스템을 개발하였다. 이를 위해 본 연구에서는 CXQuery 개념을 도입, 질의 표현식을 사 용하고 이를 XPath 로 변환함으로써 스트리밍 XML 데 이터를 효과적으로 처리할 수 있도록 하였다.

CXQuery 질의에서 데이터 이름과 조건이 주어질 때, 서브 엘리먼트, 애트리뷰트에도 그 조건을 대입시켜야 한다. 이 때, 질의를 생성하게 되면 아주 많아지게 되는데 이를 효율적으로 처리하도록 하는 방안을 연구할 예정이다.

참고문헌

[1] 이월영. “XML 데이터베이스에서 문서 구조 독립적인 질의 처리 기법”. 이화여자대학교 과학기술대학원 박사학위청구논문. 2004

[2] Yanlei Diao, Hao Zhang, Michael J.Franklin. “YFilter : Efficient and Scalable Filtering of XML Documents”. ICDE. 2002

[3] Yfilter, Berkeley University. ’http://yfilter.cs.berkeley.edu/code_release.htm”

[4] Yanlei Diao, Hao Zhang, Michael J.Franklin. “NFA-based Filtering for Efficient and Scalable XML Routing”. Technical Report, USB/CSD-1-1159. 2001

[5] Mehmet Altinel, Michael J. Franklin. ” Efficient Filtering of XML Documents for Selective Dissemination of Information”. Proceedings of the 26th VLDB Conference. 2000

[6] Yanlei Diao, Michael J. Franklin ”High-Performance XML Filtering: An Overview of YFilter”. IEEE Computer Society Technical Committee on Data Engineering. 2003

[7] W3C Consortium, XML Path Language(XPath), version 2.0, W3C Recommendation Nov 12, 2003. http://www.w3.org/TR/xpath20.html.

[8] S.Bong, D. Chamberlin, M.F. Fernanderz, D. Florescu, J. Robie, and J. Simeon, XQuery 1.0 : An XML Query Language, W3C Working Draft Nov 12, 2003. http://www.w3.org/TR/xquery/.

[9] 하비 디텔, 폴 디텔. “JAVA HOW TO PROGRAM” 5 Ed. Prentice Hall, 2003